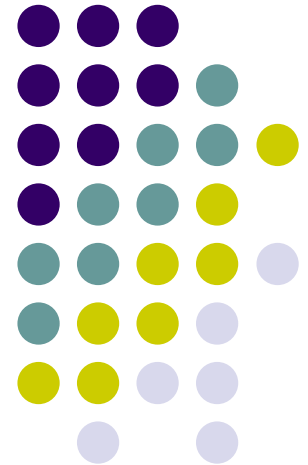


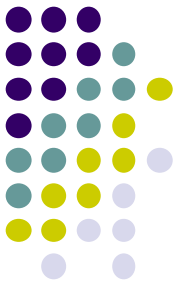
# WIX1002

## Fundamentals of Programming

---

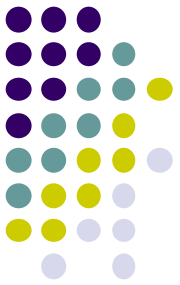
### Chapter 9 Inheritance





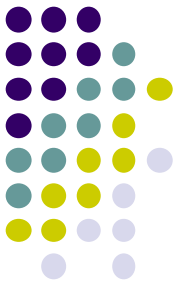
# Contents

- Introduction
- Overriding Method
- Protected
- Package Access
- Object Class



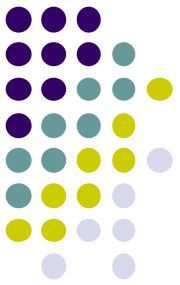
# Introduction

- One of the main techniques of OOP is known as inheritance. Advantage of inheritance is **code reuse**.
- **Inheritance** is the process by which a new class known as a **derived class** is created from another class called the **base class**.
- Inheritance is the mechanism for extending existing classes by adding methods and fields.
- The base class sometimes refer to **superclass** and the more specialized class that inherits from the superclass is called the **subclass**.
- A subclass automatically has all the instance variables, static variables and the public methods of superclass.



# Introduction

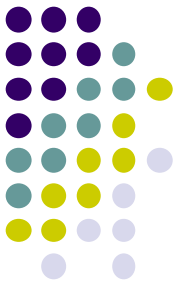
- However, a subclass has **no access to the private fields** of its superclass.
- Sometimes the base class is called the **parent class** and the derived class is called the **child class**.
- If A is the parent class of another class, it is often called **ancestor class**. If A is an ancestor of B, B is often called a **descendent** of class A.
- **super()** invoke the no-argument constructor of superclass.
- **super** can also be used to call the method of a superclass.



# Introduction

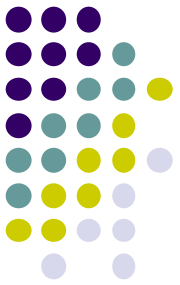
```
public class subClassName extends superClassName {  
    instance variables  
    methods  
}
```

```
public class HourlyEmployee extends Employee {  
    public HourlyEmployee() {  
        super();  
        super.superClassMethod();  
    }  
}
```



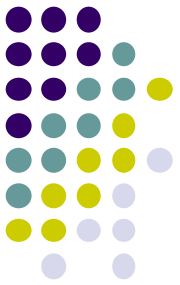
# Overriding Method

- A derived class inherits method that belong to base class. If the base class requires different definition for an inherit method, the method can be redefined or override.
- However, if the base class method consists of **final** modifier, the method can not be overridden.
- As a general rule, an override method cannot change the return type of the method definition. However, if the returned type is a class type, the returned type can be changed to the **descendant class type**.
- An override method can change the **private method** in base class **to public method** in derived class. However, it can't change the public method to private method.



# Overriding Method

```
public class BaseClass {  
    ...  
    public Employee getValue() {  
  
    }  
}  
  
public class DerivedClass extends BaseClass {  
    ...  
    public HourlyEmployee getValue() {  
  
    }  
}
```

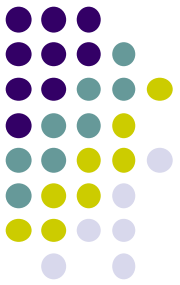


# Overriding Method

```
public class BaseClass {  
    ...  
    private void setValue(int a) {  
  
    }  
}
```

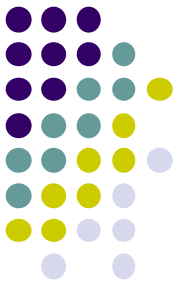
```
public class DerivedClass extends BaseClass {  
    ...  
    public void setValue(int a) {  
  
    }  
}
```





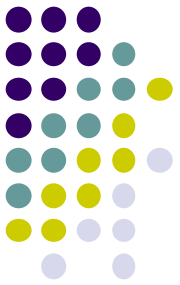
# Protected

- An object of an derived class has the type of the derived class as well as the type of the base class and every one of its ancestor classes.
- However, a derived class **can't access directly** to the **private instance variables** of the base class.
- The **protected** modifier can be used to allow the access for its own class and the derived class.



# Package Access

- A package is a namespace for organizing classes and interfaces in a logical manner. Java packages can be stored in compressed files called **JAR** files.
- If no modifiers like public, protected or private for instance variables or methods, the instance variables or methods will have **package access**.
- Package access is the default access where it can be accessed by name inside the definition of any class in the same package but not outside the package.

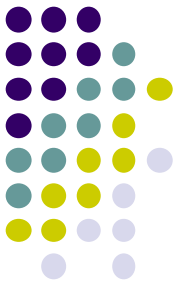


# Object Class

- In Java, every class is a derived class of the class Object.
- Every class inherit the **getClass** method from the class Object. The getClass method can be used to check the class of an object. The getClass method can't be overridden.

```
if (obj1.getClass() == obj2.getClass())
```

```
    System.out.println("The objects are belongs to same  
    class");
```



# Object Class

- The **instanceof** operator checks if an object is of type given as its second argument.
- The instanceof will return true if the object is of the type of any descendent class.

if (obj1 instanceof ClassName)

System.out.println("The object is one of the type of any descendent class of ClassName");

